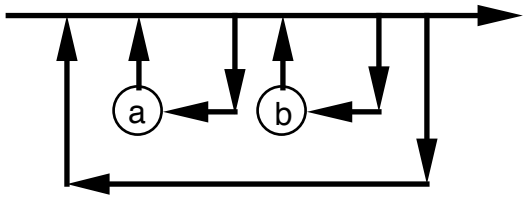
SAMPLE

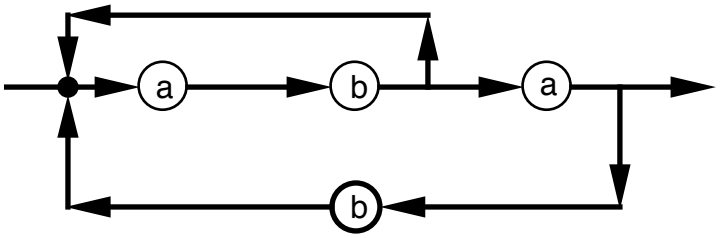
Exam I
Open book/notes

1. [20 points]
In each part of this problem, there is an extended BNF expression (terminal character set = {a, b}) *and* a syntax diagram. Determine whether or not the pair is equivalent, and either explain the equivalence, or if they are not equivalent, give an example of a string that is described by one but not the other.

(a) (a* b)*



(b) (abab)* aba



2. [25 points]
Write the Haskell definition of a function 'delSpaces' that accepts one argument that is a string (i.e., [char]), and returns a value that is a copy of the argument string with all spaces deleted. For example, delSpaces "now is the time " = "nowisthetime".

Your solution will not receive full credit without an explanation of its operation and justification that it behaves as required.

3. [25 points]

For the Haskell expression below, show its derivation tree (from <exp>):

2*x + x^3 > 0

The necessary portion of the BNF is included below. Bold **|** denotes the BNF alternative operator, bold **( )** denotes BNF grouping not expression parenthesis, and bold **[ ]** denotes the BNF "optional" operator (zero or one) -- in ordinary font, these are each Haskell characters. Syntax categories lexp and rexp are for left and right associative operators, respectively. The productions for identifiers and various types of literal constants are omitted — for this problem, assume that these syntax categories derive their familiar results in a single step.

exp → exp^0	
exp^i → exp^{i+1} **[**$qop^{(n,i)}$ exp^{i+1}**]** | $lexp^i$ | $rexp^i$	(0≤ i ≤9)
$lexp^i$ → **(**$lexp^i$ | exp^{i+1}**)** $qop^{(l,i)}$ exp^{i+1}	(0≤ i ≤9)
$rexp^i$ → exp^{i+1} $qop^{(r,i)}$ **(**$rexp^i$ | exp^{i+1}**)**	(0≤ i ≤9)
$qop^{(n,4)}$ → < | >	(non-assoc, prec 4)
$qop^{(l,6)}$ → + | –	(left assoc, prec 6)
$qop^{(l,7)}$ → * | /	(left assoc, prec 7)
$qop^{(r,8)}$ → ^ | **	(right assoc, prec 8)
exp^{10} → fexp	
fexp → **[**fexp**]** aexp	
aexp → qvar | literal	
qvar → identifier	
literal → numeral | charconst | stringconst | boolconst	

(partial) Haskell EBNF

4. [30 points]

Provide a Haskell definition of a polymorphic function 'replace' that takes three arguments — values x and y of polymorphic type 'a' and a list xs of type [a] — and returns the list xs with each occurrence of x replaced by y, and other elements unchanged. For instance, replace 's' 'l' "bass" yields "ball".

Your solution will not receive full credit without an explanation of its operation and justification that it behaves as required.